CS325: Analysis of Algorithms, Fall 2020

Practice Assignment 3 (Due: Thur, 12/03/20)

Homework Policy:

1. Students should work on practice assignments individually. Each student submits to CANVAS one set of *typeset* solutions in pdf format.
2. Practice assignments will be graded on effort alone and will not be returned. Solutions will be posted.
3. The goal of the assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
4. You are allowed to discuss the problems with others, and you are allowed to use other resources, but you *must* cite them. Also, you *must* write everything in your own words, copying verbatim is plagiarism.
5. More items might be added to this list. ☺

Problem 1. For each of the following statements, respond *True*, *False*, or *Unknown*.

(a) If a problem is decidable, then it is in P.

(b) For any decision problem, there exists an algorithm with exponential running time.

(c) $P = NP$.

(d) All NP-complete problems can be solved in polynomial time.

(e) If there is a reduction from a problem A to Circuit SAT, then A is NP-hard.

(f) If problem A can be solved in polynomial time, then A is in NP.

(g) If problem A is in NP, then it is NP-complete.

(h) If problem A is in NP, then there is no polynomial time algorithm for solving A.

Problem 2. A k-CNF formula is a conjunction (AND) of a set of clauses, where each clause is a disjunction (OR) of a set of exactly k literals. For example,

$$(a \vee b \vee c \vee \neg d \vee \neg e) \wedge (\neg a \vee b \vee c \vee \neg x \vee \neg y) \wedge (\neg x \vee y \vee c \vee d \vee a)$$

is a 5-CNF. The k-SAT problem asks if a k-CNF formula is satisfiable. In class we saw that 3-SAT is NP-hard. In contrast, 2-SAT is polynomially solvable.

(a) Show that 4-SAT is NP-Complete (For partial credit in an exam, specify all the statements you need to conclude that 4-SAT is NP-complete even if you cannot prove them).

(b) Describe a polynomial time algorithm to solve 1-SAT.

Problem 3. Suppose you are given a magic black box that can determine in polynomial time, given an arbitrary graph G, whether G is 3-colorable. Describe and analyze a polynomial-time algorithm that either computes a proper 3-coloring of a given graph or correctly reports that no such coloring exists, using the magic black box as a subroutine. [Hint: The input to the magic black box is a graph. Only a graph. Vertices and edges. Nothing else.]

Extra problems. Don't submit the following problems, but try to solve them!

Problem A. Consider the following problem, called BoxDepth: Given a set of n axis-aligned rectangles in the plane, how big is the largest subset of these rectangles that contain a common point?

(a) Describe a polynomial-time reduction from BoxDepth to MaxClique.

(b) Describe and analyze a polynomial-time algorithm for BoxDepth. [Hint: $O(n^3)$ time should be easy, but $O(n \log n)$ time is possible.]

(c) Why don't these two results imply that P=NP?

Problem B. The problem 12-coloring is defined as follows: Given an undirected graph G, determine whether we can color each vertex with one of twelve colors, so that every edge touches two different colors. Prove that 12-coloring is NP-hard. [Hint: Reduce from 3-coloring.]